# **Revision**

## Week 1

### Day 3

* Intro to C#
  + What's compile time and run time?
  + What is .NET?
  + What is the difference between .NET Framework, NET Core, and .NET5/.NET6?

### Day 4

* Using Visual Studio
  + What is a conditional breakpoint?
    - A breakpoint that only triggers if certain conditions are met
  + What is the difference between compiling to debug and release formats?

### Day 5

* Unit Testing & Refactoring
  + What is unit testing?
    - Testing individual (smallest) units of functionality, e.g. single methods
  + Why should we write unit tests?
    - Cheap to write, save bugs being caught at a higher (more expensive) level, easily automated, can be used to assert acceptance.
  + How can you decide what test input to use?
    - Case by case
    - partition testing and edge value testing are useful in ranged behaviour
    - As are empty list, single element list, negative values, out of range values
  + What is a TestCase? Why use it instead of separate tests
    - Allows us to carry out multiple tests on a single function (save time on writing each separately).
    - DRY.
    - Easy to read and check for (reasonable) completeness.
  + What is refactoring?
    - Applying a refactoring
    - Rewriting code so that it is easier to read and maintain without changing its functionality
  + Why did we need to refactor the Main method in yesterday's example?
* Test First Development

## Week 2

### Day 6

* Operators
  + Give an example of a ternary operator
    - grade = mark >= 65 ? “Pass” : “Fail”
  + What does the modulus operator do?
    - Returns the remainder of a division operation
  + What is the result of dividing integers 10 by 4
    - 2 (value does not carry decimal section)
  + How would you calculate the number of weeks and days represents by 34 days?
    - Int weeks = 34 / 7
    - Int days = 34 % 7
  + What is the difference between a postfix and a prefix increment operator?
    - Postfix – x++ – assigns then increments
    - Prefix – ++x – Increments then assigns
  + What is the difference between a unary and binary operator?
    - Unary operators take one operand (argument), e.g. x++
    - Binary operators take two operands, e.g. x + y
  + Which types of operator have the lowest priority?
    - Assignment and lambda operations
* Selection
  + What is the difference between an if-else block and a switch-case block? Give an example of when you would use each.
    - If-else has more flexibility as it tests a complete expression (e.g. ranges)
    - Switch blocks require specific values (e.g. int values, string values, enums)
* Iterations
  + What are the 4 types of loop (just name them) –
    - for
    - foreach
    - while
    - do while
  + What is the difference between a for- loop and a foreach- loop? Give an example of when you would use each.
    - For loop creates a numerical index which is good when positional information is useful, can be used to assign, can go through a collection in reverse order.
      * Would use when assigning calculated values to a pre-sized array.
    - Foreach loop passes through each item in order without consideration of size, but items are read-only.
      * Would use when using (but not altering) values in another calculation (e.g. mean, mode)
  + What is the difference between a while- loop and a do while- loop? Give an example of when you would use each.
    - A while loop checks the condition at the beginning of the loop, so it may not even run once.
      * A loop to interpolate for or flag patchy data may not need to run if there isn’t any.
    - A do while loop will always run once as it checks the condition at the end.
      * Console input needs to check for validity after it has been done at least once.

### Day 7

* Exceptions
  + In handling exceptions, when is the finally block run?
    - Always – Unless there is an unhandled exception that interrupts execution.
* Numerical Data Types
  + What happens if you add 3 to the largest int? To the largest unsigned int?
  + What is the smallest floating point number type?  How many bytes?
    - Float (4 bytes)
  + What data types can int (Int32 [4 bytes]) be safely converted to?
    - long (Int64 [8 bytes]), float [4 bytes], double [8 bytes] or decimal [16 bytes]
  + How would you find out the largest value an Int32 value type can be?
    - Check the documentation
    - Work it out from memory usage
    - Check Int32.MaxValue
  + What is casting?
    - Explicitly changing from one datatype to another
  + How can you prevent silent overflow of integers?
    - Use the “checked {}” block

### Day 8

* More Types – Strings
  + What is StringBuilder and why is it used?
    - String builder creates mutable strings that can be operated on in situ. It cuts down on the number of calls required to accomplish certain goals. It does not have all methods.
  + What is string parsing?
    - The reading of information in a string into another data type
* More Types – Arrays
  + What is the difference between a multidimensional array and a jagged array?
    - A multidimensional array is a 2, 3 or n-dimensional data table.
    - A jagged array is an array of arrays
* More Types – Date Time
  + Why is representing Dates and Times complicated?
    - Because they aren’t base 10
    - Because months have different numbers of days
    - Because different locations have different date formats

### Day 9

* More Types – Enums
  + What is an enum and why would you use it?
    - Enumerated constants (either auto-enumerated or set by the developer)
    - Useful in situations where a selection of specific options exists, rather than ranges or an assortment of letters.
* Methods
  + What makes up a method signature?
    - Access modifiers (public, private, protected, internal)
    - (not) static -> static methods
    - Return type (data type)
    - Name (PascalCase)
    - Arguments
  + What is a method body?
    - The code that constructs the functionality of the method
  + In a method signature, what does the void keyword mean?
    - No return value
  + What is method overloading?
    - Having two or more methods with the same name but different parameter lists to allow for multiple approaches to input
  + What is an out parameter?
    - Specifies a variable should be passed by reference making it accessible outside the method scope.
  + What are named parameters and why are they useful?
    - e.g. ( doThing(isPizza: true) will be read into the isPizza parameter
    - Allows specific identification of parameter inputs rather than relying on positional information
    - Allows parameters to be specified out of order
  + What is the difference between passing method parameters by value and by reference?
    - By value passes a copy of the value and that copy does not affect the original
    - By reference passes the heap reference that allows the original to be accessed and changed.
  + What types are normally passed by value? By reference?
    - By value – Primitives such as numerical types
    - By reference – collections such as arrays, lists or strings
  + What is a tuple, and why is it useful?
    - Immutable collection of same/different kinds of information denoted by () that can allow for the packaging of multiple objects for a simple return.
* Memory Model
  + What is a stack, what is a heap?
    - Stack – Fast access memory for storing/retrieving value data or references to objects on the heap – Cleared of scope items when they go out of scope. Last In First Out (LIFO).
    - Heap – Slower memory for storing larger objects (e.g. strings, lists). Items are only live while they are being referred to by an item on the stack.
  + When does the .NET garbage collector run? What does it do?
    - Every so often
    - Flags dead objects in memory for removal
    - Removes them
    - Compresses the rest of the items to optimize remaining memory (defrag).
* Four Pillars and their Implementation in C#
  + What is abstraction?
    - A user doesn’t need to understand an engine to know how to drive
  + What are the elements of a C# class?
    - Signature, Data (Private fields, Public Properties), Constructors, Methods
  + What is encapsulation?
    - Keeping the internal data and methods hidden
  + How does C# support encapsulation?
    - Access keywords – e.g. private, protected
  + What does a constructor do?
    - Creates an instance of the class
  + What is the difference between a constructor and other methods?
    - Has no return type (returns an instance)
  + What is a C# property?
    - A wrapper for a private field with get and set methods

* What is object initialisation?
  + Setting variables as part of the constructor call using {}
* What does the static keyword mean?
  + Attached to the class not to an instance of the class
* What is a backing field?
  + The private field that a property hides
* What does the virtual keyword mean?
  + Implemented but overridable
* What happens if you attempt to override a method that is not marked virtual?
  + Red squigglies?
* What does the sealed keyword mean?
  + Method: Inheritors of that method cannot change it
  + Class: A class cannot be inherited

* What is an abstract method? An abstract class?
  + An abstract method is one that is not implemented, but is left for implementation in derived classes. An abstract class can hold abstract and concrete methods.
* What is an interface? Can an abstract class be instantiated?
  + A contract for the implementation of a series of methods. No, an abstract class cannot be instantiated.
* Does an Abstract class have to have Abstract methods? Describe a scenario where you would use an abstract class.
  + No. I would use it if I wanted a partial implementation, e.g. just ToString but wanted derived classes to implement their own methods (e.g. area of a shape)
* What is the difference between method overloading and method overriding?
  + Overloading is having more than one method with the same name to allow multiple sets of input parameters
  + Overriding is replacing an inherited method with a newly implemented one.
* What are the similarities and differences between classes and structs?
  + Structs cannot inherit or be inherited from
  + Structs represent a single object (e.g. DateTime) whereas classes represent a type of an object.
  + Classes instantiate reference types, structs are value types
* What class is the base class for all C# classes?
  + Object
* What properties and methods does the Object class have?
  + ToString, Equals, GetHashCode
* What does the Object ToString() method do by default?
  + Returns namespace and class

Object Equality and Comparison

* How would you override the Equals method?
  + Two functions – one to cast to the correct type, one to deal with the correct type
* If you override Equals, what other method do you need to override and why?
  + GetHashCode because Equals uses it to compare
* How would you implement the CompareTo method of the IComparable interface?
  + < 0 this instance before obj
  + 0 this instance same place as obj
  + > 0 this instance after obj
* What is the relationship between CompareTo and Equals?
  + CompareTo is for Sorting. Equals is for Equality?

Collections

* What is the best Collection type for fast access of sequentially stored items?
  + Linked lists?
* What is the difference between a Stack and a Queue?
  + FIFO vs LIFO
* What other Collection types are available? Briefly describe them.
  + Heap
  + Dictionary (Key, Value) pairs
  + ArrayList (deprecated) – Hetrogenous objects
  + HashSet
  + Linked Lists (don’t implement ILIst – no index)

SOLID

* What are the 5 SOLID principles?
* Describe the Single Responsibility Principle
  + A class should have only one reason to change
* Describe the Open/Closed Principle
* Describe the Liskov Substitution Principle
* Describe the Interface Segregation Principle
  + Many small interfaces rather than a monolith
* Describe the Dependency Inversion Principle

Advanced NUnit

* What is the difference between the NUnit Framework and NUnit3TestAdapter?
* Give some other examples of NUnit test runners
* What does the attribute [SetUp] indicate and when does the associated method run?
* If an NUnit class has a constructor, when does it run?
* What is the major difference between the NUnit framework and MSTest?
* What are the three As of unit testing?
* What are the 5 FIRST characteristics of good unit tests?
* How do you exclude a test from being run?
* What does the [TestCaseSource] attribute do?
* What is the TDD cycle?
* What are the advantages of using TDD?
* Advantages of TDD :
  + You only write code that’s needed
  + More modular design
  + Easier to maintain
  + Easier to refactor
  + High test coverage
  + Tests document the code
  + Less debugging
* What are the pitfalls of doing TDD?
  + No silver bullet
  + slow process
  + All the members of a team got to do it
  + Tests got to be maintained when requirements change

## Week 5

### Efficiency and Recursion

* What is a recursive function?
  + A function that calls itself working down to a base case where it tops
* What is Big-O notation?
  + “A measure of algorithmic complexity”
  + Ignore constants means O(30n) is regarded as the same as O(n)
  + A representation of the worst-case time complexity of an algorithm
* Give an example of an O(n) process. O(1)? O(n squared)?
  + O(n) – Linear search through an array
  + O(1) – Dictionary, a simple operation
  + O(n^2) – A Loop within a loop
* Why are some algorithms considered unreasonable?
  + Because their time requirements are so huge that they could not be calculated in a reasonable amount of time

### General OOP

* What is OOP and why should we use it?
  + Object-Oriented-Programming
  + Modularity for troubleshooting (and testing)
  + Reuse of code through inheritance
  + Flexibility through polymorphism
  + Models the Real-World (problem solving)
* What is the difference between Encapsulation & Abstraction?
  + Encapsulation – Keep implementation details inside (don’t mess with the car’s engine)
  + Abstraction – Rely on an interface rather than having to manipulate methods directly (A car can “drive” we don’t need to know how it does it)

### Refactoring and Code Smells

* Define the term "Refactoring"
  + Changing the implementation (easier to maintain, easier to read, faster to run) of code without changing it’s behaviour
* When should you refactor your code?
  + Red-Green-Refactor
  + Rule of 3
* When should you NOT refactor your code?
  + Just before deadlines
  + When it needs a complete rewrite
* Define the term "Code Smells"
  + Common problematic structures or behaviours in code
  + “A sign that you should refactor your code”
* Name some Code Smells
  + Data clump, Long method, Shotgun Surgery, Inappropriate Intimacy
* What is a data clump? How should you refactor your code if you find one?
  + Data clump – Bunches of data that hang around together
  + **Extract Class**
  + Preserve Whole Object
  + Introduce Parameter Object
* What is feature envy and what should you do about it?
  + Feature Envy – When a method makes too many calls to other classes to obtain data
  + Extract Method
  + Move Method
  + Move Field
* If a class has several methods which contain if/else or switch blocks that test a "type" attribute, what should you do about it?
  + Use polymorphism and make each type a polymorphic object

### Design Patterns

<https://www.freecodecamp.org/news/the-basic-design-patterns-all-developers-need-to-know/>

* What is a Design Pattern?
  + A known solution to a common programming problem
* What problem does the Singleton Design Pattern try to solve?
  + Prevent multiple creation of classes where this would be unneeded or undesirable (e.g. Database Links, Loggers, Entity Framework).
* Describe how the Singleton Design Pattern could be implemented in C#
  + A single globally available class
  + Accessible through ‘static’, ‘\_instance’
  + Constructor is made ‘private’
* What problem does the Factory method pattern attempt to solve? How is it implemented in C#?
  + Define an interface for creating an object, but let subclasses decide which class to instantiate. Factory Method lets a class defer instantiation to subclasses.
  + Factory Class – Pass in identifier
* What is the Strategy Design Pattern?
  + Algorithm varies independently from the client that uses it
    - e.g. the “Shoot” method for a Camera, Hunter, Weapon in the polymorphic shootout
* What is the Decorator Design Pattern?
  + Wrapper Class implementing an interface that is inherited by a subclass(es)
  + “Attach additional responsibilities to an object dynamically. Decorators provide a flexible alternative to subclassing for extending functionality.”
* What are the 3 categories of Design Patterns?
  + Creational – Singleton
  + Behavioural – Strategy
  + Structural – Decorator
* What is MVC?
  + Model – Data
  + View – What the user sees
  + Controller – Communicates with both the Model and the view

JSON and XML

* Describe the JSON structure
  + Key-Value pairs, Data focusses
* What C# collection does JSON correspond to?
  + Dictionary
* What data type are the keys in JSON ?
* Describe the XML structure
  + Between Tags. Have to open and close
* What is the difference between XML and HTML?
  + XML is extensible, HTML has defined tags
* Why are JSON and XML used?
  + JSON – To transfer data – Focussed on content
  + XML – With formatting

### Files, Logging and Streaming

* What are the options for logging during program execution?
  + Debug – Only shows up in Debug mode
  + Trace – Shows up in Debug and Release mode
* Why would you want to use conditional compilation?
  + Multiplatform code that needs to be excluded
  + Test only code
* What is the difference between a text and a binary file?
  + Both in bits
    - Text files
      * map bits to characters
      * use a text encoding such as ASCI
      * human readable
      * newlines…has EOF
    - Binary files can encode other types (custom data)
      * e.g images
      * not human readable
      * no newlines… no specific EOF
* What is a file signature?
  + Data used to identify or verify the contents of a file, e.g. what kind of file is it? Image, Sound, etc.

What is streaming?

* What do all .NET streams have in common? Why are there different types of derived streams?
  + They inherit the Stream object
  + To deal with different encoding/decoding or transmission protocols

SQL

* What is an ERD?
  + Entity Relationship Diagram – Describes the Entities (Tables) in the database
* What is a primary key?
  + A unique identifier for a particular row in the table
  + Cannot be NULL
* What is a foreign key?
  + An identifier that references the primary key in another table
* What is the difference between a foreign and primary key?
  + Primary keys must be unique
* Explain Normalisation.
  + Making your database Tables match with the definitions of the Six (and more) Normal forms (especially 1-3)
  + Most improving efficiency and removing redundant data
* What command would you use if you want to add a table?
  + CREATE TABLE (*variables*);
* What command would you use if you want to add a column to a table
  + ALTER TABLE *table* ADD *column* **type**, *column* **type**
* What command would you use if you want to delete a row of data
  + DELETE FROM *table* WHERE *row\_mathcing\_condition*
* What command would you use if you want to insert data into a table
  + INSERT INTO (*list\_of\_columns*) VALUES (*data\_list\_in\_same\_order\_as\_list\_of\_columns*)
* What are DML, DDL, DCL and TCL?
  + DML – Data Manipulation Language (data in the table) [INSERT, UPDATE, DELETE,…]
  + DDL – Data Definition Language (structure of the table) [CREATE, DROP, ALTER,…]
  + DCL – Data Control Language (rights and permissions) [GRANT, REVOKE]
  + TCL – Transaction Control Language (transactions within the database) [COMMIT, ROLLBACK, SAVEPOINT, SET TRANSACTION]
* If you wanted to select only the top 10 highest earners in a table of millionaires, what query would you write?
* What is a wildcard?
  + A character that can replace one or more character in WHERE *variable* LIKE *string*
  + Examples are:
  + “\_” for one character
  + “%” for zero or more characters
  + “[ABC]” for a character group
  + “[^ABC]” to exclude a character group
* What is concatenation (with example)
  + Joining strings first\_name + ‘ ‘ + last\_name (or || in place of ‘+’)
* What query would I write if I want to find out all customers who do not have city listed in a table
* What does the arithmetic operator % do?
  + Modulo – The remainder after a division
* What does NULL mean?
  + A missing value
  + Nothing is equal to NULL
* Can a primary key be NULL?
  + No
* How do I select all the values starting with the letter C?
  + SELECT *values* WHERE *values* LIKE ‘c%’;
* When is the GROUP BY keyword used?
  + In aggregates

## Week 6

### Lambda Expressions

<https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/lambda-expressions>

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/delegates/delegates-with-named-vs-anonymous-methods>

<https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/delegate-operator>

* ENTITY FRAMEWORK: How do you load associated objects in a query?
* ENTITY FRAMEWORK: Why isn’t eager loading enabled by default?
* What is a Lambda expression? Why is it used in LINQ queries?
  + A declaration of an anonymous function (declared in place, cannot be reused, parameters inferred from context)
  + They are used in LINQ queries as delegate types or expression trees may be used.
* What does x => x \* x mean?
  + x goes to x which is multiplied by x
* What is an anonymous method?
  + Defined using the **delegate** keyword
  + Defining an inline function with the option of taking a parameter
* What is Expression body syntax?
  + A body to the right of the => and returns the result of the expression
* When is a LINQ query executed
  + Whenever the IEnumerator is called

## Async Programming

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/async/>

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/async/async-return-types>

* Why do we need to use asynchronous methods?
  + API Calls
  + Web requests/services
    - If the outcome of a method is dependent on a service that takes time
    - To improve the execution speed of tasks that can be run in parallel.
* What keywords should an asynchronous method use (and where?)
  + \_ \_ **Task** NameWith**Async**()
* What return types are allowed for asynchronous methods?
  + Task, Task<T>, void (event handler), object (with GetAwaiter), IAsyncEnumerable<T> (returns an Async stream)
* What effect does the await keyword have?
  + Waits for a task to be complete
  + Returns to the caller method while waiting
* What is the naming convention for asynchronous methods?
  + **\_\_\_\_Async()** in the method name
  + Use of **async** and **await** keywords

## Serialisation

<https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/serialization/>

* What is serialisation?
  + Converting an object into a stream of bytes to store the object or transmit it to a database, memory or a file
* What are the advantages and disadvantages of using binary object serialisation? JSON Serialisation? / XML Serialisation?
  + Binary – Fast and compact for use in storage and socket-based network streams. BinaryFormatter cannot be secured and shouldn’t be used for trusted data.
  + XML –
  + JSON –

### API and REST

* What does API stand for?
  + Application Programming Interface
* What are the HTTP verbs and what are their CRUD equivalents?
  + POST - Create
  + GET - Read
  + PUT - Update
  + DELETE - Delete
* What is the structure of an HTTP request? An HTTP response?
  + Request: request line, headers, empty line, body (optional)
  + Response: status code, headers, empty line, body (optional)
* What are the categories of HTTP response status code?
* <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>
  + 1xx – Informational responses
  + 2xx – Successful responses
  + 3xx – Redirection messages
  + 4xx – Client error responses
  + 5xx – Server error responses
* What does REST stand for in the context of RESTful APIs?
  + REpresentational State Transfer
* What are the characteristics of a REST API?
  + Stateless – Client and Server understand messages without needing to know about previous messages
  + Separation of Client and Server – Only the message structure matters, client and server can be set up or changed independently of each other.
  + Specifies resource using URI
* For the endpoint myapi.com/api/customers what would you expect a GET request to do? A POST request? PUT? DELETE?
  + GET – List of customers or a specific customer
  + POST – Create a new customer/customers
  + PUT – Update customer information
  + DELETE – Remove a customer from the database
* What do we mean by caching?
  + Local storage of query items so they can be used quickly and without having to send extra requests for them
* What should the RESTful endpoint myresource.io/Employees/6/Order/2 GET?
  + Employee Id 6 with order Id 2
* Give some examples of header elements that can be used to control caching
  + max-age, no-cache, no-store

### RestSharp

* What does the RestSharp library do?
  + HTTP REST API client library for .NET
  + Automatic serialization/deserialization
  + Request and response type detection
  + Authentication and other features
* What does a testhosts.dll.config file do? Why is it useful?
  + Loads settings for the app using ConfigurationManager
* Why did we create separate data handling and call manager folders in our Restsharp test framework?

### Test Doubles using Fakes and Moq

* What are the 5 types of test doubles?
  + Dummy, Fake, Stubs, Spies, Mocks
* What does a Fake do?
  + Implemented but with shortcuts
* Why should you use an in-memory database for testing?
  + In-Memory means it is controllable and fast
  + Testing against a real database replicates the actual functionality related to the test (either as an environment or as the object under test)
* Why should you use the Moq framework for testing?
  + Can isolate a class under test by mocking its dependencies and verifying its calls
* How does Dependency injection aid unit testing?
  + We can use doubles for dependencies far more easily
* When would we want to use a dummy?
  + To fill a parameter list
  + A dummy is an object that is passed around but never used
* What does a Stub do?
  + Provides canned answers to expected calls.
  + No responses to anything outside the test
* How can we use a Moq to check if a method is called with the correct parameters?
  + .Verify to check if a particular method has been called
* What is the difference between strict and loose mocking behaviour?
  + Loose mocks don’t have to implement every method on an object. Strict mocks do.
* What is returned if a method is not set up (loose)
  + Doesn’t throw exceptions
  + Returns default values (pr empty arrays, enumerables, etc.)
* What is returned if a method is not set up (strict)
  + Throws exception

Requirements Traceability Matrix (RTM)

* Forward – Maps requirements to test cases
  + Confirms – Test coverage for requirements, Project trajectory
* Backward – Maps test cases to requirements
  + Avoids – Scope Creep
* Bi-Directional Traceability – Both of the above in a single document
  + Establishes each requirement has relating test cases

## Week 6 DEV – 129

### Test Doubles using Fakes and Moq

* What are the 5 types of test doubles?
  + Dummy, Fake, Stubs, Spies, Mocks
* What does a Fake do?
  + Implemented but with shortcuts
* Why should you use an in-memory database for testing?
  + In-Memory means it is controllable and fast
  + Testing against a real database replicates the actual functionality related to the test (either as an environment or as the object under test)
* Why should you use the Moq framework for testing?
  + Can isolate a class under test by mocking its dependencies and verifying its calls
* Why did we refactor our WPF-EF system to use a Service class?
  + The service class can be isolated from the business logic
  + Both can be tested as separate components
* How does Dependency injection aid unit testing?
  + We can use doubles for dependencies far more easily
* When would we want to use a dummy?
  + To fill a parameter list
  + A dummy is an object that is passed around but never used
* What does a Stub do?
  + Provides canned answers to expected calls.
  + No responses to anything outside the test
* How can we use a Moq to check if a method is called with the correct parameters?
  + .Verify to check if a particular method has been called
* What is the difference between strict and loose mocking behaviour?
  + Loose mocks don’t have to implement every method on an object. Strict mocks do.
* What is returned if a method is not set up (loose)
  + Doesn’t throw exceptions
  + Returns default values (pr empty arrays, enumerables, etc.)
* What is returned if a method is not set up (strict)
  + Throws exception

### API and REST

* What does API stand for?
  + Application Programming Interface
* What are the HTTP verbs and what are their CRUD equivalents?
  + POST - Create
  + GET - Read
  + PUT - Update
  + DELETE - Delete
* What is the structure of an HTTP request? An HTTP response?
  + Request: request line, headers, empty line, body (optional)
  + Response: status code, headers, empty line, body (optional)
* What are the categories of HTTP response status code?
* <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>
  + 1xx – Informational responses
  + 2xx – Successful responses
  + 3xx – Redirection messages
  + 4xx – Client error responses
  + 5xx – Server error responses
* What does REST stand for in the context of RESTful APIs?
  + REpresentational State Transfer
* What are the characteristics of a REST API?
  + Stateless – Client and Server understand messages without needing to know about previous messages
  + Separation of Client and Server – Only the message structure matters, client and server can be set up or changed independently of each other.
  + Specifies resource using URI
* For the endpoint myapi.com/api/customers what would you expect a GET request to do? A POST request? PUT? DELETE?
  + GET – List of customers or a specific customer
  + POST – Create a new customer/customers
  + PUT – Update customer information
  + DELETE – Remove a customer from the database
* What do we mean by caching?
  + Local storage of query items so they can be used quickly and without having to send extra requests for them

### API Development with ASP.NET

* What do we mean by scaffolding a Controller? What files are added to the project?
  + Using the Model items Context and Item to generate code for the Controller
* What does ASP stand for and how long has it been around? What makes it active?
  + Active Server Pages
  + 1996
  + Actively created using data held by the server
* How does the structure of an ASP.NET application relate to MVC?
  + Models are the Model
  + Controllers are the Controller
  + For an API the View is the JSON object
* What actions happen on startup of an ASP.NET application?
  + Sets up the builder
  + Adds the Services to the DI
  + Sets up the middleware pipeline
* What are the responsibilities of a Dependency injection container?
  + <https://docs.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-6.0>
  + <https://www.martinfowler.com/articles/injection.html>
  + “Recording, deciding and settling dependencies”?
  + “Determining, creating and injecting all dependencies”?
  + ? – Hold the Service objects
  + ? – Serve up the Service objects when requested
* What are the 3 classes of Service Lifetimes? What do they mean?
  + Transient – New service every time it is called
  + Scoped – New service with each request
  + Singleton – New service on start (of the program and lasts for the lifetime of the program)
* What are the advantages/disadvantages of adding a service to the DI container with Singleton lifetime?
  + Advantage: no need for continuous creation/garbage collection
  + Disadvantage: memory expensive
* What happens in the request pipeline? (middleware?)
  + Passes request through from top to bottom serving aspects of the request
  + Passes to controllers for request to be turned into response
  + Passes response back through from bottom to top checking on the way out
* Name some components of the request pipeline
  + HTTP Redirection, Authorisation, Authentication, Map to Controllers
* Why is Exception handling normally the outermost layer of the pipeline?
  + <https://docs.microsoft.com/en-us/aspnet/core/fundamentals/error-handling?view=aspnetcore-6.0>
  + “*The developer exception page runs early in the middleware pipeline, so that it can catch unhandled exceptions thrown in middleware that follows.*”
* Which comes first in the request pipeline - Authorisation or Authentication?
  + <https://docs.microsoft.com/en-us/aspnet/web-api/overview/security/authentication-and-authorization-in-aspnet-web-api>
  + Authentication – Logging in
  + Authorisation – Can the logged in user do what they are trying to do?